SYNTAX ZX80®

A PUBLICATION OF **THE HARVARD GROUP**

VOL.2 NO.6 ISSN 0273-2696 JUN., 1981

CAI UPDATE

CAI Instruments is almost ready with their Widget series peripherals for ZX80s and MicroAces. One small problem remains. The Widget series tape drive, printer and Widget board were compatible with the original Sinclair 8K ROM. Like most of us, CAI has only just gotten their new 8K ROM and now must check that their peripherals do not interfere with the functions of the new ROM. According to Bob Swann of CAI, this should delay production only about two weeks. Swann also announced that CAI is developing home financial software to keep budget and income tax records. Prices are not definite, but they should come in under $20. Some programs will be ready in June.

NEWS FROM MICROACE

MicroAce and ZX80 owners who want the flicker-free feature of the ZX81 can get it with an add-on board from MicroAce. The board contains only 6-7 chips and solders onto your PC board. The cover fits over the machine with the board installed, although the fit is a little tight on a ZX80. According to Chris Cary of MicroAce, the add-on board will cost $29.50 and be available by June. Contact MicroAce at 1348 E. Edinger, Santa Ana, CA, 92705, 714/547-2526.

MicroAce's Santa Ana shop will expand to sell other personal computer brands, including the PMC, VIC-20, PET, and TRS-80 Color Computer.

MICROPERIPHERAL ABANDONS MODEM

Microperipheral Corp. of Mercer Island, WA, has given up its plans to produce ZX80 modems, according to Lucy Stoner of Microperipheral. Because they had to add many hardware features not in the ZX80, she said, the modem would have cost more than the original computer.

EZUG NEWSLETTER

The Educational ZX80/1 Users' Group (EZUG) publishes a newsletter each month for ZX80 users. The British EZUG newletter covers programs and ideas from its membership for educational uses of the ZX80 and ZX81 computers. Just send a self-addressed stamped envelope for delivery. (The copy we saw would cost 40¢ for airmail postage.) SYNTAX has arranged an information exchange with Eric Deeson, organizer of EZUG, to bring you ideas from overseas. For more information, contact Eric at Highgate School, Balsall Heath Rd., Birmingham, England, B12 9DS.

The 8K ROM produces a different screen display than the 4K ROM. SYNTAX will publish listings in 4K format, but we will print both 4K and 8K Syntactic Sums for programs that run in either machine.

Rochester Institute of Technology offers two intensive computer courses for deaf adults in August. Contact Donald Beil, NTID Data Processing Dept., RIT, 1 Lomb Memorial Dr., Rochester, NY, 14623, 716/475-6373, voice or TTY.

SYNTAX ERRORS

May 81--Marty Iron's Line Renumbering program contained an error. Line 9130 should read: 9130 IF PEEK(S)=248 THEN GO TO 9300.
Our list of new 8K ROM features should have included the INT(X) function.

4K PROGRAMS ON 8K ROM

By now you know your 8K ROM won't load 4K tapes. So far, all you can do is retype the program.

Why so? The 8K system looks for file headers, or leading name fields (you must name files), even using LOAD "" (load first program found). A 4K tape lacks headers, so the 8K system can't tell where 4K programs begin. If you loaded a 4K program, your 8K ROM would read it using the wrong character codes. For example, code 18 means minus to 4K ROMs and greater than to the 8Ks. And the keyword codes also changed. Your 8K tapes hold programs and system variables; 4K tapes won't set these properly.

Even if you overcome these difficulties, you'll find your old programs don't run if they depend on integer arithmetic, logic true being -1, the TL$ function, or the masking properties of the 4K ROM's logical functions.

Is the situation hopeless? Only until someone works out how to program the translation between notations. The 4K and 8K BASIC are nearly compatible. A routine diverting LIST to a 4K machine bus via a translation table may, with interfacing, send 4K programs to the bus of a second 8K machine in 8K codes. Of course, this plan needs 2 ZX80s.

Or perhaps someone can invent a program for 4K machines to write simulated 8K tapes. Or a program to read a 4K tape on 8K machines.

We'll publish solutions as we learn of them.

BEGINNERS' COMPUTING MAGAZINE

OnComputing, a quarterly computer magazine, is going monthly and changing its name to Popular Computing. The new magazine will be for "intelligent readers who lack a technical background" and will be edited exclusively for beginners. Christopher Morgan, founding editor of onComputing and editor-in-chief of BYTE, will remain as editor of Popular Computing. Contact onComputing, Inc., 70 Main St., Peterborough, NH, 03458.

OUR POLICY ON CONTRIBUTED MATERIAL

SYNTAX ZX80 invites you to express opinions related to the ZX80 and the newsletter. We will print, as space allows, letters discussing items of general interest. Of course, we reserve the right to edit letters to a suitable length and to refuse publication of any material.

We welcome program listings for all levels of expertise. Programs can be for any fun or useful purpose. We will test run each one before publishing it, but we will not debug programs; please send only workable listings.

In return for your listing, we will pay you a token fee of $2.00 per program we use. This payment gives us the nonexclusive right to use that program in any form, world-wide. This means you can still use it, sell it, or give it away, and so can we.

We will consider submissions of news and hardware or software reviews. Please keep articles short (350-400 words). Again, we reserve the right to edit accepted articles to a suitable length. We will pay 7 cents per 6 characters, including spaces and punctuation, for accepted articles.

When you send in programs for possible publication in SYNTAX, please include the following information:

- How to operate the program, including what to input if it does not contain prompts.
- Whether you can run the program over again and how.
- How to exit the program.
- The Syntactic Sum (using the Syntactic Sum program in the February, 1981, issue).
- Whether it fits in 1K or 2K RAM (or 16K when available).
- Whether it uses the 4K or 8K RAM.

We pay for this explanatory text at the same rate as for articles in addition to payment for the program itself.

If you want us to return you original program listing or article, please include a self-addressed, stamped envelope. Otherwise, we cannot return submitted material.

SYNTAX ZX80 is published monthly by Syntax ZX80, Inc., a wholly-owned subsidiary of The Harvard Group, RD 2, Box 457, Bolton Road, Harvard, MA 01451. Telephone 617/456-3661. Subscriptions: 12 issues, $25. Single issue, $4.

Publisher: Kirtland H. Olson
Editor: Ann L. Zevnik
Associate Editor: Susan G. Barber
Technical consultant: Eric K. Olson

Printed by Joseph E. Marhefka, Jr.
Clinton Offset Printers
Clinton, MA 01510

 ISSN 0273-2696

INVERSE CRYPTOLOGY

This challenging code breaking game for 2 uses 4K ROM and 1K RAM. The computer encodes the secret phrase typed in by player 1. Player 2 guesses by typing letter=letter. A correct guess replaces the coded letter with the correct inverse letter.

DIM (25) stores the code. Subscripts 0 to 25 represent the real alphabet. The program recodes the phrase each time it prints it instead of storing a separate coded phrase. It's slower but uses less memory.

Enter up to 1 line. To use up to 84 characters, delete line 425 and change line 410 to PRINT "ENTER LETTER=LETTER". To exit, break the code or hit SHIFT BREAK quickly after a guess. To play again, hit RUN twice, then (NL).

Harold A. Lamkin, Mt. Clemens, MI

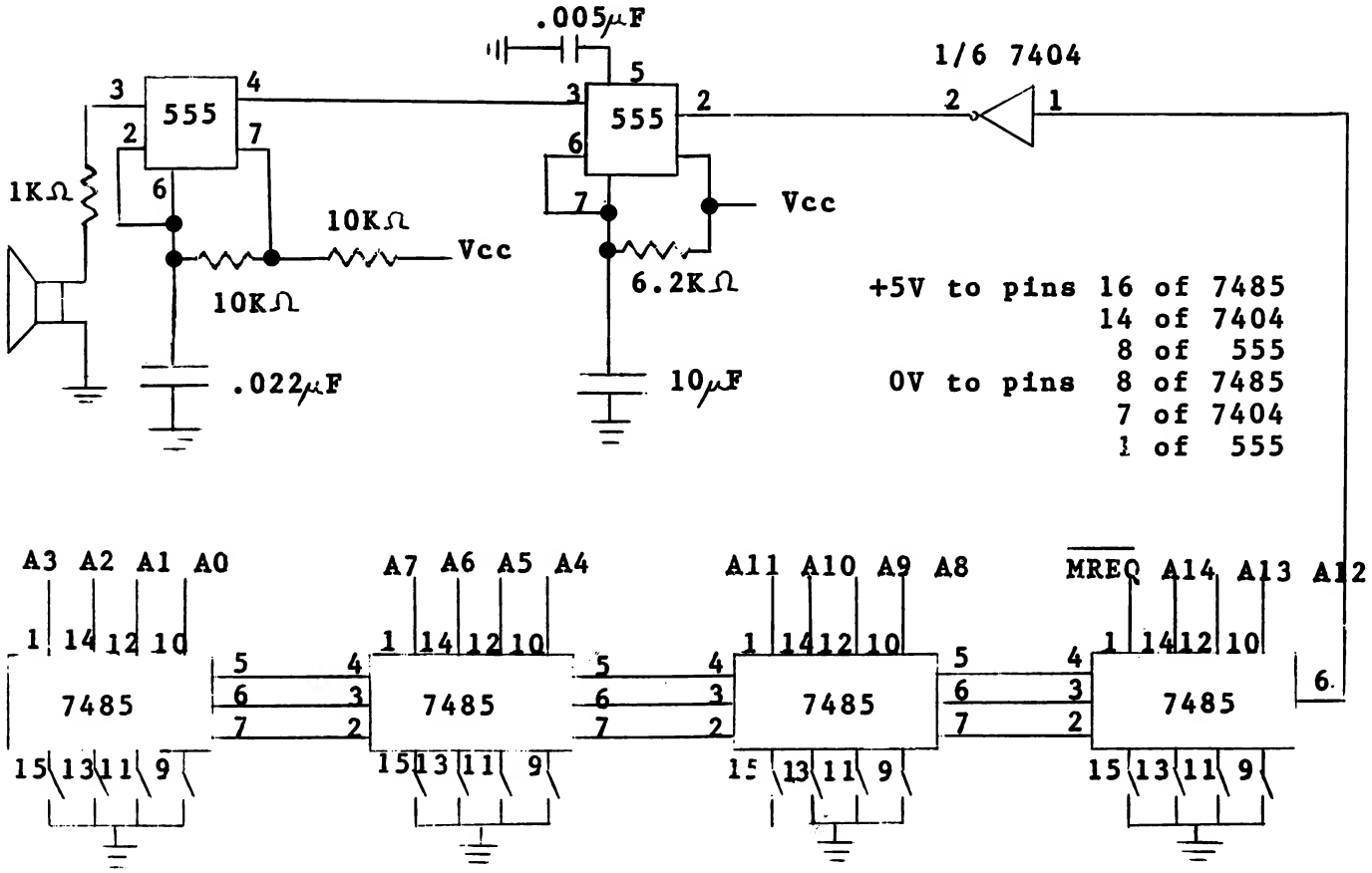
```
  10 DIM A(25)
  20 RANDOMISE
  30 FOR Z=0 TO 25
  40 LET A(Z)=RND(26)+37
  60 FOR X=0 TO 25
  70 IF A(Z)=A(X) AND NOT Z=X TH
EN GO TO 40
  80 NEXT X
  90 NEXT Z
 100 PRINT "STATE A PHRASE"
 110 INPUT W$
 120 GO SUB 600
 400 PRINT
 410 PRINT "ENTER PHRASE OR LETT
ER=LETTER"
 420 INPUT U$
 425 IF U$=W$ THEN GO TO 900
 430 IF NOT CODE(TL$(U$))=227 TH
EN GO TO 700
 440 LET Y=CODE(TL$(TL$(U$)))
 450 IF Y<38 OR Y>63 THEN GO TO
700
 460 IF NOT A(Y-38)=CODE(U$) THE
N GO TO 700
 470 LET A(Y-38)=Y+128
 480 CLS
 485 PRINT "CORRECT"
 490 GO SUB 600
 491 IF Z=0 THEN GO TO 900
 495 GO TO 400
 600 PRINT
 605 LET Z=0
 610 LET U$=W$
 620 LET Y=CODE(U$)
 630 IF Y<38 OR Y>63 THEN PRINT
CHR$(Y);
 640 IF Y<38 OR Y>63 THEN GO TO
650
 645 PRINT CHR$(A(Y-38));
 646 IF NOT A(Y-38)=Y+128 THEN L
ET Z=Z+1
 650 IF Y=1 THEN GO TO 680
 660 LET U$=TL$(U$)
 670 GO TO 620
 680 PRINT
 690 RETURN
 700 LET X=X+1
 705 CLS
 710 PRINT "TRY AGAIN"
 720 GO SUB 600
 730 GO TO 400
 900 CLS
 910 PRINT "YOU BROKE THE CODE W
ITH ";X-26;" ERRORS ";W$
Syntactic Sum=-7191
```

ALL-PURPOSE BEEPER

Here's a nifty circuit to emit a "beep" every time your ZX80 or MicroAce reads from or writes to a selected memory address. Four comparators, the heart of the circuit, constantly compare the address on the bus to the address set on a bank of 16 switches. When the two are equal, a 555 timer is started, which runs another 555 oscillator. The tiny switches in the circuit must be set in binary, one digit at a time. The switch which would be used for A15 is left always grounded instead and compared to NOT MREQ. The schematic this page shows how to construct this circuit. Use a 2 inch 10 ohm speaker to emit the beep.

Now you can hear every time the computer executes a certain byte of code. Depending on what you set the switches to, you can hear the beep for nearly any occurence within your computer.

To get a beep when	Set switches as
Newline key pressed	0283
Keystroke ignored (NL with error, etc.)	03A7
Keystroke results in syntax error	08AE
Character printed on the screen	4024
Printing on screen advances to a new line	4025
Rubout pressed	0395
Screen is cleared (By any method)	0747
Keystrokes in line beginning with illegal keyword	07BB



MACHINE LANGUAGE PROGRAMMING--
MULTIPLICATION AND DIVISION

Last time we tried Sinclair's machine language (ML) routine for determining available memory. It added 11 to the stack pointer and returned an answer to BASIC--easy stuff. Subtraction in binary is also easy; subtract column-by-column and borrow when necessary.

Multiplication and division require more thought and attention to detail. One or two commands will not multiply 2 numbers, as in BASIC. Still, ML multiplication and division routines are straightforward. If you don't understand some terms here, don't worry--just follow the logic and use the program as a recipe.

Refer to the books by Rodnay Zaks (Programming the Z80, 2d ed. pp. 113-136) and Kathe Spracklen (Z-80 and 8080 Assembly Language Programming, pp. 60-65) for more details. Zaks shows sample code and Spracklen gives math examples.

Glance through this assembly listing that converts degrees Fahrenheit to Celsius using the formula C=5(F-32)/9. (This problem would be easier to solve using BASIC, but we'll use ML to see how it's done). The first column is the memory address where beginning bytes of instructions reside. If the instruction needs more bytes, they will be stored in successive locations. The second column lists hexadecimal program codes (you can enter these using Matthew Johnson's Hex Monitor, May 81). Column three labels subroutines--

marking parts of the program. Assembly language commands are in the fourth and fifth columns. The last column contains comments, preceded by semicolons. Like REM statements, comments identify functions of the lines.

Note the repetitive look of commands in column five. Good ML programming breaks problems into small tasks the computer can do.

In decimal multiplication, you shift each partial product 1 column to the left for each left shift you make in the multiplier:

```
(multiplicand)        32
(multiplier)        x 15
(partial product)    160
(partial product)     32
(product)            480
```

Mentally, you do 1x32, and shift 1 place left to multiply by 10.

Look closely and you'll see the routine shifts the multiplier in register C to the right. Shift either way, but stick to one direction throughout the problem, as in this decimal example:

```
SHIFT LEFT          SHIFT RIGHT
    23                  23
  x 12                x 12
    46                  23
   23                   46
   276                 276
```

In binary multiplication all partial products are either the multiplicand or a row of zeros, since 1 times any number is that number and 0 times any number is 0. (Examine the multiplication, and remember, 1+1=10 in binary. Carry the 1 to the next column, as in decimal addition.)

At the end of SUBT (subtract), we have taken the Fahrenheit value from a POKEd memory address (see PEEK-POKE story this issue), subtracted 32, and left the multiplicand (F-32) in the E half of the DE register pair, a temporary Z80 storage location.

MULMR loads the multiplier (5) and sets a bit-counter in register B (again, a temporary storage location) that we count down with each shift. Z80's use a special command, DJNZ PROCESS, that decrements B and jumps to the named process if B is not 0. When B is 0, the Z80 knows it's done.

MULT multiplies binary numbers bit-by-bit, as you multiply decimal numbers column-by-column. MULT shifts the multiplier right into the carry bit and adds the result to the HL register each time the carry bit is 1. The loop continues until the bit-counter reads 0 (at the other end of the multiplier) meaning all bits have been counted. Meanwhile, the HL register pair accumulates a running partial product total.

```
 C-register  carry bit
   11001101  0   (beginning)
 0>01100110  1   (incr prod & shift)
 0>00110011  0   (shift only)
```

Dividing is like multiplying, except that the divisor (number you divide by) is subtracted from the dividend (number you divide into). DIV performs binary long division after DIV66 loads the divisor (9) into D, zeros E and resets the bit-counter, B. Each time the computer subtracts divisor from dividend, it adds 1 to the quotient and shifts the new dividend and quotient left 1 bit.

The following examples show how long division is the same in both decimal and binary:

```
DECIMAL                    BINARY
    103                          1100111
21 2163                  10101 10000111011
   21↓↓                        10101↓
    063                         11001
    63                          10101↓
     0                           100100
                                  10101↓
                                   11111
                                   10101↓
       quotient                    10101
divisor dividend                   10101
                                       0
```

If the subtraction yields a negative result, the program must restore the quotient total and the

dividend to their previous values prior to proceeding. Finally the answer is in L and the remainder in H; the next-to-last command zeros the remainder.

This routine returns integer quotients to BASIC. Use C9 as the last command to get back to BASIC from machine language.

You can convert Fahrenheit values from 32 to 491 degrees. Beyond these limits, the program fails because the result in the L register is outside its 0-255°C range. Another program must handle negative or larger results.

To run the program, you need to POKE a Fahrenheit value from BASIC. For 32-255°F, POKE 17357, val of F and POKE 17358,0. For 256-491 F, POKE 17357,val of F-256 and POKE 17358,1. After POKEing the values, use PRINT USR(17359) to see the result.

You can write a whole BASIC program to use this ML program. First, enter the memory protect program, p.8. Use 1 for K and 51 for N. This saves 51 bytes at the top of memory from encroachment from BASIC programs. DO NOT TYPE NEW ON 4K ROMS. Load Hex Monitor and change the starting address to be 17407 (top of 1K RAM) minus the bytes you reserved, 51, in line 90. Input the hex values for the conversion program. Then load or type in a BASIC program to ask for Fahrenheit values, POKE the ML memory locations, call the USR routine at 17407-51+2 (to avoid the 2 bytes where you POKEd in the F values) and display the results.

ASSEMBLY LISTING (4K only; see p.12)

```
43CD             00080          ORG     43CDH
43CD 0000        00090 TEMP     DEFW    0
43CF 2ACD43      00100 SUBT     LD      HL,(TEMP)       ;LOAD FAHREN
43D2 1600        00110          LD      D,0             ;CLEAR D
43D4 1E20        00120          LD      E,20H           ;LOAD SUBTRAHEND
43D6 A7          00130          AND     A               ;CLEAR CARRY
43D7 ED52        00140          SBC     HL,DE           ;FAHRENH-32
43D9 EB          00150          EX      DE,HL           ;RESULT INTO DE
43DA 210000      00160          LD      HL,0            ;CLEAR HL
43DD 0E05        00170 MULMR    LD      C,5H            ;LOAD MULTIPLIER
43DF 0608        00180          LD      B,8H            ;INITIALIZE COUNTER
43E1 CB39        00190 MULT     SRL     C               ;SHFT RGT TO CARRY
43E3 3001        00200          JR      NC,NOADD        ;TEST CARRY
43E5 19          00210          ADD     HL,DE           ;(FAHRENH-32)+RESULT
43E6 EB          00220 NOADD    EX      DE,HL
43E7 29          00230          ADD     HL,HL           ;DOUBLE SHIFT
43E8 EB          00240          EX      DE,HL
43E9 10F6        00250          DJNZ    MULT            ;LOOP UNTIL B=0
43EB 1609        00260 DIV66    LD      D,9H            ;LOAD DIVISOR
43ED 1E00        00270          LD      E,0             ;CLEAR E
43EF 0608        00280          LD      B,8H            ;INITIALIZE COUNTER
43F1 A7          00290 DIV      AND     A               ;CLEAR CARRY
43F2 ED52        00300          SBC     HL,DE           ;DIVID-DIVIS
43F4 23          00310          INC     HL              ;QUOT=QUOT+1
43F5 F2FA43      00320          JP      P,NOA22         ;TEST POSITIVE
43F8 19          00330          ADD     HL,DE           ;RESTORE IF NECESS
43F9 2B          00340          DEC     HL              ;QUOT=QUOT-1
43FA 29          00350 NOA22    ADD     HL,HL           ;DOUBLE SHIFT
43FB 10F4        00360          DJNZ    DIV             ;LOOP UNTIL B=0
43FD 2600        00370          LD      H,0             ;SET RMNDR ZERO
43FF C9          00380          RET
```

PLACING USR CALLS IN 4K, 8K

Now that our readers use both 4K and 8K ROMs, SYNTAX supports both sets of users. We'll tell you how to overcome differences in the 2 designs. Here's how to put ML at the same addresses in both.

With the 8K ROM, USR routines reside at high RAM addresses. You POKE the first address you want to use to locations 16388 & 16389 (p.168 of ZX81 manual). Use the following program to get equal space in 4K machines so SYNTAX ML programs will run at specified locations. Use of ROM routines and system variables or locations must change, but many routines work in either machine.

Answer the first prompt with the number of K of memory on your machine. Respond to the second prompt with the number of bytes you want reserved. Next load your BASIC program; it will be kept away from the ML area. (The usual program would aid loading ML, say a monitor.)

Your first user location will then be at 16382-N+(1024*K).

```
  5 INPUT K
  6 INPUT N
  7 LET M=16384+(K*1024)-1-N-1
 10 POKE 17152,33
 20 POKE 17153,M-(M/256)*256
 30 POKE 17154,M/256
 40 POKE 17155,195
 50 POKE 17156,107
 60 POKE 17157,2
 70 LET X=USR(17152)
```

Syntactic Sum= 11550,4K

Although this program sets aside space at the top of any size memory, we will publish programs assembled for 1K memories. To use our code directly, respond to the first prompt with 1.

You input your memory size and the space you want for USR routines. This BASIC program then loads HL with the address of a fake top of RAM and jumps to the INIT routine at 26BH, 619 dec (see ROM listing) after the computer would have set the memory top.

4K ROM LISTING—INITIALIZATION

In the 4K ROM listing below, E8LINE means E-LINE.


```
0261             00070          ORG     0261H
400A             00071 E8LINE:  EQU     400AH
4028             00072 RAMBOT:  EQU     4028H
4008             00073 VARS:    EQU     4008H
4000             00074 Y:       EQU     4000H
                 00075 ;INIT:   ENTER HERE ON RESTART WITH HL=7FFFH, A=3FH
0261 3601        00076 LI2:     LD      (HL),1  ;SET LOCS 4000H TO 7FFFH TO 1
0263 2B          00077          DEC     HL
0264 BC          00078          CP      H
0265 20FA        00079          JR      NZ,LI2  ;LOOP UNTIL HL=3FFFH
0267 23          00080 LI3:     INC     HL      ;SET ALL RAM LOCS TO ZERO
0268 35          00081          DEC     (HL)
0269 28FC        00082          JR      Z,LI3   ;LOOP UNTIL WRAPAROUND OR RUN OUT
026B F9          00083          LD      SP,HL   ;ASSUMES 1ST ROM BYTE IS NOT 01H
026C F5          00084          PUSH    AF      ;MARK "NO RETURN BLOCKS"
026D 3E0E        00085          LD      A,0EH
026F ED47        00086          LD      I,A
0271 ED56        00087          IM      1
0273 FD210040    00088          LD      IY,Y
0277 212840      00089          LD      HL,RAMBOT
027A 220840      00090          LD      (VARS),HL;NO LINES OF PROGRAM
027D 3680        00091          LD      (HL),80H ;VARIABLES CONSIST ONLY OF
027F 23          00092          INC     HL      ;TERMINATOR RECORD
0280 220A40      00093          LD      (E8LINE),HL;LINE TO BE EDITED STARTS NEXT
```

ZX80 PRESTEL MODEL

This program is a model of the British Prestel data base tree structure. It can be used with your own data for display to show how Prestel data is accessed. This version fits in 1K; if you have 16K and lots of patience, you could create and record an impressive dictionary in a given field.

From EZUG Newsletter, No. 3, pp. 7-8, Apr/May 81, Highgate School, Balsall Heath Rd, Birmingham, B12 9DS, UK. Reprinted by permission.

(Note--Prestel is an on-line data base retrieval system run by the British Post Office. Meant for the public, it uses phone lines and home TV sets and resembles the US Source network in theory. This program is complete for choice 1. Choices 2, 3, & 4 are only place holders; they contain no data yet. Try this format to create sequentially accessed data, as in choice 1. Prestel doesn't fit in 1K RAM with 8K ROM, so we give no 8K Syntactic Sum.)

```
  10 LET A=0
  20 GO SUB 5000
  30 LET A$= "TYPE NUMBER NEEDED
"
  40 PRINT A$
  50 GO SUB 5100
  60 PRINT "1. V, I, R"
  70 PRINT "2. CIRCUIT EQUATION"
  80 PRINT "3. POWER"
  90 PRINT "4. RESISTORS"
 100 GO SUB 5100
 110 PRINT "OR TYPE 0 TO STOP"
 120 INPUT B
 130 CLS
 140 IF B=0 THEN STOP
 150 IF B<0 OR B>4 THEN GO TO 10
 160 GO TO B*1000
1000 LET A=1
1010 GO SUB 5100
1020 PRINT A$
1030 GO SUB 5100
1040 PRINT "1. TO GET V"
1050 PRINT "2. TO GET I"
1060 PRINT "3. TO GET R"
1070 GO SUB 5100
1080 INPUT B
1090 CLS
1100 IF B<1 OR B>3 THEN GO TO 10
00
1110 GO TO 1100+B*100
1200 LET A=11
1210 GO SUB 5000
1220 PRINT "USE V=IR"
1230 GO SUB 5200
1300 LET A=12
1310 GO SUB 5000
1320 PRINT "USE I=V/R"
1330 GO SUB 5200
1400 LET A=13
1410 GO SUB 5000
1420 PRINT "USE R=V/I"
1430 GO SUB 5200
2000 PRINT 2000
2010 GO SUB 5200
3000 PRINT 3000
3010 GO SUB 5200
4000 PRINT 4000
4990 STOP
5000 FOR C=2 TO 11
5010 PRINT " ";
5020 NEXT C
5030 PRINT "PAGE ";A
5040 GO SUB 5100
5050 RETURN
5100 FOR C=1 TO 3
5110 PRINT
5120 NEXT C
5130 RETURN
5200 GO SUB 5100
5210 PRINT "N/L TO RETURN TO IND
EX"
5220 INPUT C$
5230 CLS
5240 GO TO 10
5250 RETURN
```

Syntactic Sum=-16780,4K

PROGRAMMED RESPONSES--LATIN ROOTS

This program is a drill for students of Latin and Greek. The computer displays the Latin and Greek root and the student types in the correct English version.

Use this technique to write your own programmed drills. Any drill, like math or languages, with predictable answers is suit-

able. Try adding lines to ask student's name and use it during the drill. Pete Cone's multiplication program (Dec.80) is a good example. Or add PRINT statements rewarding correct answers.

Daniel Ambrosini, Claverton, NY

```
 10 PRINT "AG,AC",
 15 INPUT A$
 20 IF NOT A$="DO" THEN GO TO 1
0
 25 PRINT "AGR",
 30 INPUT A$
 35 IF NOT A$="FARM" THEN GO TO
 25
 40 PRINT "CAD,CAS",
 45 INPUT A$
 50 IF NOT A$="FALL" THEN GO TO
 40
 55 PRINT "CANT",
 60 INPUT A$
 65 IF NOT A$="SING" THEN GO TO
 55
 70 PRINT "CAP,CEP",
 75 INPUT A$
 80 IF NOT A$="TAKE" THEN GO TO
 70
 85 PRINT "CAPIT",
 90 INPUT A$
 95 IF NOT A$="HEAD" THEN GO TO
 85
 100 PRINT "CEDE",
 105 INPUT A$
 110 IF NOT A$="GO" THEN GO TO 1
00
 115 PRINT "CELER",
 120 INPUT A$
 125 IF NOT A$="SPEED" THEN GO T
O 115
 130 PRINT "CLUD,CLUS",
 135 INPUT A$
 140 IF NOT A$="CLOSE" THEN GO T
O 130
 145 PRINT "CUR",
 150 INPUT A$
 155 IF NOT A$="RUN" THEN GO TO
145
 160 PRINT "DICT",
 165 INPUT A$
 170 IF NOT A$="SAY" THEN GO TO
160
 999 PRINT "END"
Syntactic Sum=-30558,4K 34142,8K
```

HARDWARE REVIEW: KEYBOARD BEEPER

Burnett Electronics' keyboard beeper consists of a tiny piezoelectric disk attached to two integrated circuit chips on a little circuit board. After installation, the disk emits a soft beep whenever you depress a key. The entire apparatus fits inside a ZX80 or MicroAce case.

All you need to wire the circuit into your computer is a soldering iron and the ability to connect 12 wires. The circuit is already internally wired. As the rankest of beginners in electronic circuit building, I was elected to test the device and its written directions. I successfully installed my beeper the first time in 20 minutes with no trouble. Burnett's directions are exceptionally clear and easy to follow, even if you don't know a resistor from a diode.

However, the installation instructions did cause me two slight problems: first, one wire was tough to distinguish because the directions identified it only by color, and it looked just like the other white wire nearby. Second, my electronics experts had to tell me to tape over the solder side of the beeper circuit board before taping it to my ZX80 case to avoid shorting it, a precaution Burnett did not mention. (This does not matter with an unshielded MicroAce, which has no metallization inside the case.)

The circuit works perfectly and produces variable tones as long as you hold the key down. The computer case gives it enough resonance to make the sound loud enough to hear but not be annoying. If you have trouble telling when you make contact with the keys, this beeper can help you type more quickly and accurately. Keyboard beeper, $12.00, Burnett Electronics, 908 Morris St., Cincinnati, OH, 45206.

DEAR EDITOR:

In your article on new 8K ROM features, several, such as READ-DATA-RESTORE, LEFT$, RIGHT$, MID$, DRAW-UNDRAW, INT(X), were not on your list. What happened to these very important functions? Did you forget to put them on your list? Has Sinclair phased them out?

David Shulman, Peabody, MA

In redesigning the 8K ROM for their printer, Sinclair eliminated several planned features, including most of those you mention. LEFT$ and RIGHT$ were never intended. We left INT(X) out of the list; it is a feature of the 8K ROM. You can get around the lack of LEFT$ and RIGHT$ by using the LEN(X) function. For LEFT$(A$,N), just use A$(1 TO N). For RIGHT$(A$,N), use A$(LEN(A$)-N+1 TO LEN(A$)). Replace MID$(A$,N,M) with A$(M TO N).--AZ

Like all ZX80 users, I have struggled with the imprecision of the action of the pressure-sensitive keyboard. Finally I came up with a solution. I cut a set of small clear circles out of a soft plastic sheet, like the top of a refrigerator food storage container, using a paper punch. I then affixed 1 on each key with double-stick Scotch tape. The raised circles guide my fingers to the key centers and direct the pressure to the contacts. I can see the letters through the circles, but for the first time I can actually touch-type with a marked increase in speed and accuracy.

Julian Alexander Jr.,
Scotch Plains, NJ

I understand the ZX80's video blanking during program execution reflects the designer's preference for speed over continuous display. Is it possible, via user software or simple firmware, to reallocate some of the Z-80A's speed to support a video display while running a program? Second, is there a technically oriented manual or documentation for the ZX80 produced by Sinclair or others that is available here or in England?

James Janner Jr., Seattle, WA

Apparently it is possible. Check our review of Softsync's ZX80 Invasion game in May's SYNTAX. The author accomplished continuous video display using machine language programming, but we haven't figured out how yet. Also, check our story on p.1 for a hardware source, MicroAce's add-on boards for ZX80s and MicroAces.

Try LINSAC's book, The ZX80 Companion, (available from Image Computer Products, 615 Academy Dr., Northbrook, IL 60062, 312/564-5060)) as a ZX80-specific technical information source. Others may exist, but I haven't seen them yet.--AZ

Can the basic circuit from Build Additional RAM (Mar.81) be used to build even larger RAM modules? I would like to build a 32K version, but I need additional information.

Charles Losinski, Winona, MN

The ZX80/MicroAce can accomodate up to 48K RAM externally, according to David Ornstein of Sinclair. The Z80A can address 64K locations, but the ROM uses 16K. If you write to the empty ROM addresses, you'll get a write only memory--the computer will instantly forget what you wrote. To build 48K, just bus all data lines together, then expand the chip select scheme to decode the extra addresses, depending on what size of RAM chips you add.--AZ

I built the interface using the 8212 (Interface to the Real World, Jan.81) and find my address different from the one in the article (28672-32767). I checked my address lines going to the interface and found them to be correct. Does the MicroAce differ from the ZX80 in this respect?

Gerald Johnson, Redwood City, CA

Bill Clark, MicroAce technician, says the only difference between the 2 ROMs is that data pins 3 & 4 have been swapped. Otherwise, the 2 ROMs are identical and are software compatible, meaning that the same software will work on both. They are not hardware compatible, however; you can't put a MicroAce ROM in a ZX80 because of the switched data lines. So your addresses should be the same whether you use a ZX80 or a MicroAce.--AZ

Here's an improvement to Joe Chaiet's Bar Graphics program (Apr.81) for 2K MicroAces. Change these lines:

```
 130 IF U(Y)<X THEN PRINT CHR$(
128);
 133 IF Y<10 AND T=1 THEN PRINT
 "▄▄";
 134 IF Y<10 AND T>1 THEN PRINT
CHR$(128);CHR$(128);
 135 IF Y=10 AND T=1 THEN PRINT
 "▄";
 136 IF Y=10 AND T>1 THEN PRINT
CHR$(128);
```

Lines 175 and 181 need a ; at the end. Also, DIM(50) need only be DIM(10).

R. Bruce Hosken, Space Coast Microcomputer Club, Merritt Island, FL

Will the new LLIST, COPY, and LPRINT statements (on Sinclair's 8K ROM) control CAI Instruments' peripherals?

Bill Cotham, Winston-Salem, NC

According to Bob Swann of CAI, they may not. As reported in the story on p.1, CAI had to revamp their peripherals design to work with the new 8K ROM. They are trying to make the Sinclair printer commands work on their printer, but they may have to use different ones.--AZ

These SYNTAX readers would like to hear from others in their areas. If you would like to contact local ZX80/MicroAce users, send us your name and address. We'll publish them when space permits.--AZ

*John M. Morrison, 327 Mayland Ave., Morrestown, NJ, 08057, 609/234-0230.
*Steven K. Stroh, 4316 Avenue R:, Galveston, TX, 77550.
*Bob Childress, 399 Fremont St., San Francisco, CA, 94105, 415/421-7845.
*Bob Irwin in Houston, TX, would like to hear from other ZX80/MicroAce users interested in forming a ZX80 special interest group on Compuserve. Contact Bob on Compuserve, 70315,123.

MODIFYING HEX MONITOR FOR 4K

Hex Monitor (May 81) stores ML programs without shielding them from BASIC. These changes load ML to reserved space (program p.8).

In 90 LET S=dec val of start address of hex code.

In 270, change USR(S) to USR (dec val of start of computation).

If you encounter error 4, increase the number following the equal sign in line 200 (display line number where Monitor will clear the screen and continue).

On 4K ROM machines, USR (argument) returns the contents of the HL register pair or the argument if HL is not altered.

With 8K ROMs, USR (argument) returns the contents of the BC register pair. Also, see the restrictions on page 167 of the ZX81 8K manual.

8K SYNTACTIC SUM

To create the 8K version of Program 2, key in the program from the Feb.81 issue. RUN the program and key in the numbers from this new decimal listing:

```
 33 125  64 237  91  12  64 221
 33   0   0 124 186  32   8 125
187  32   4 221 229 193 201  78
  6   0 221   9  35  24 236
```

When you have typed in the final number, error 9/5 appears. Get back to the listing (LIST) and delete lines 1-5. SAVE the rest of the program to tape.

If you use more than 1K of RAM, use the table to alter these instructions and Program 2. Always POKE 16388,224 first, regardless of memory size.

MEM	POKE 16389,	M=	USR()
1K	67	1024	17376
2K	71	2048	18400
4K	79	4096	20448
xK	67+4(x-1)	x*1024	16352+M

To use Syntactic Sum with the 8K ROM and 1K RAM, do these steps:

- POKE 16388,224 and POKE 16389,67 (This replaces Program 1)
- NEW
- LOAD the 8K version of program 2 from tape
- Type GOTO 6 (NL).
- LOAD or key in the BASIC program to be summed.
- PRINT USR(17376)

Syntactic Sum™ Syntax ZX80 Inc. and The Harvard Group.

4K SYNTACTIC SUM MODIFICATION

Run Syntactic Sum in less RAM; change line 20, program 1 to:

```
20 POKE 17153,M-(M/256)*256
```

BYTES AVAILABLE

To see how many bytes of RAM your ROM uses up, type NEW (NL),then one of these lines:

(4K ROM) PRINT 17408-PEEK(16400)-PEEK(16401)*256 (we get 947)

(8K ROM) PRINT 17408-PEEK(16412)-PEEK(16413)*256 (we get 815)

8K ROM PROGRAM CHANGES

Our 8K ROM programs published in January were written on the original 8K ROM. To run Chart Your Biorhythms on the 8K ROMs now available, change these lines:

```
 109 LET L=6
 230 LET B=N(2)-N(1)
 290 PRINT AT L,0;B=N(1)-N(2)+1;
AT L,(SIN (P2*B/23)*W+W+K);"P";A
T L,(SIN (P2*B/28)*W+W+K);"E";AT
 L,(SIN (P2*B/33)*W+W+K);"I";AT
L,(W+K);":"
```

These changes also correct typos in the published version.

MATRIX DRAWING

This 4K ROM program draws characters on a 13 by 20 matrix. To operate, type RUN. When the computer expects input, enter a number (1-13), indicating which line you want a character on. The second input (1-20), designates the column. The third input is the character code of the character you want at the designated location. For 8K, change all POKEs to write to spare memory.

Robert Ryan, Gladstone, OR

```
   1 LET A=19999
   6 LET V=0
  10 FOR I=1 TO 13
  20 FOR J=1 TO 20
  30 IF V>0 THEN GO TO 50
  40 POKE A+(I*20)+J,128
  50 PRINT CHR$(PEEK(A+(I*20)+J)
);
  60 IF J=20 THEN PRINT " "
  70 NEXT J
  80 NEXT I
  90 INPUT D
 100 INPUT B
 110 INPUT C
 120 POKE A+(D*20)+B,C
 130 CLS
 131 CLEAR
 135 LET V=1
 137 LET A=19999
 140 GO TO 10
Syntactic Sum=18662,4K
```

PROGRAMMING LANGUAGE TERMINOLOGY

You've read about BASIC, machine language (ML) and assembly language programming in SYNTAX. What are these languages?

BASIC (which stands for Beginners' All-purpose Symbolic Instruction Code) is a high-level language. This means that you do not communicate directly with the machine in its native language, but in a more complicated one (that is, more complicated to the computer). Programs in BASIC go through a BASIC interpreter, built into the ROM, to change them into machine-readable code. BASIC is an easy language for beginning programmers because the commands resemble simple English words. Programs in BASIC execute more slowly than those in lower level languages because they are broken down into more machine language commands than BASIC commands you started with. Every time you hit (NL), for example, the BASIC interpreter converts that single instruction into 7 ML commands.

Machine language is just that--the language the computer understands directly. The computer thinks only in ones and zeros, so this language consists solely of binary representations of commands and data.

Because most of us don't think in binary, we have assembly language as a way to write programs that are easier for both the computer and us to understand. Assembly language uses symbols to stand for memory addresses, commands and data. You can enter assembled programs yourself using a program like Hex Monitor (May 81) or you can run it through an assembler, a program that translates assembly language into machine language that the computer can read directly. Unlike BASIC programs, assembly symbols translate one-to-one into ML commands.

PEEK, POKE & CHR$ FOR BEGINNERS (4K)

Most of the BASIC commands on your ZX80/MicroAce are pretty clear--SAVE, LOAD, GO TO and RUN do just what they say they do. PEEK and POKE are a little tougher to understand, but as you'll see, they also do just what they say they do. CHR$ just translates the computer's character codes into the characters themselves.

As we found last month, your computer stores everything in memory locations, named by unique addresses. The ZX80 requires an address to understand a PEEK command, so all PEEK commands are of the form: PEEK(address). This tells the computer to look in the specified address. Adding a PRINT command before PEEK will make it display what it finds there.

Typing in PRINT PEEK(address) (NL) as an immediate mode command (without a line number) so the machine executes it right away, gives an answer on the screen that is a number from 0 to 255. Your computer stores characters in 8 bits, or 1 byte, so 255 is the largest number it can hold in 1 location. These numbers correspond to the character code set found on pp.75-78 of the ZX80 4K manual and pp.55-56 of the MicroAce manual. For example, if the answer is 38, that is the character representation in the address you PEEKed. If you are in a BASIC area, the character stored at that address is A. If it's a machine language area, 38 will be a ML command or data. PEEK looks to see what's in the computer's memory at a given address.

To try this out, clear your computer by typing NEW (NL) and type 9876 PRINT A*B. PEEK addresses in the immediate mode in order starting with the first address for BASIC programs, 16424. (Type with no line number: PRINT PEEK(16424), then PRINT PEEK (16425), and so on.) Don't start

with the first RAM address, 16384, because you will see system variables which won't make any sense to you. Look up the answers you get in the character code table. They will translate to PRINT A*B because you PEEKed the beginning of memory where that line resides. The first 2 bytes, or addresses, won't translate to the line number for reasons explained below.

The computer can also translate the character code for you. The BASIC command CHR$ will change the code into a character according to the character code table when you write it in the form CHR$(number) and the number is in the range 0-255.

Putting these 2 ideas together, let's write a program to look into memory and tell us what character, token or keyword is there. Set up a FOR-NEXT loop to increment from the starting address as many as you would like to see (10 in the example below). Then use the CHR$ command to change the PEEKed code into characters and keywords:

```
10 FOR I=0 TO 10
20 LET S=PEEK(16424+I)
30 PRINT 16424+I,S,CHR$(S)
40 NEXT I
```

Line 30 prints each address, character code and code translation. The commas separate the answers into columns for easy reading. You'll see code where line numbers should be, but they won't form the line numbers. This is because the ZX80 stores line numbers as pure binary code, not as character codes. When it tries to translate the binary, it comes up with garbage using CHR$. To translate them yourself, multiply the first byte's code by 256 and add the second. In our program, the first 2 codes are 0 and 10, and (0x256)+10=10, or our first line number. Change all the line numbers to 1000s (1000, 2000, etc.) and the first 2 codes are 3 and 232. (3x256)+232=1000. Try this for different line numbers.

POKE, on the other hand, puts something into an address. POKE commands are of the form: POKE address,number. The address is where you want the computer to put the number after the comma. This number should also be in the range 0-255 to represent character code or machine language code.

To our original PEEK program, now add:

```
50 POKE 16426,10
60 PRINT PEEK(16426),CHR$(PEEK(
16426))
```

The computer prints the same information as before. Line 50 puts a new character code into the first address of the program after the line number (all line numbers use up 2 addresses). Line 60 prints what the computer now finds at that address and what character that code translates into. When you go back to the program listing, you'll see that the translation for 10 (a graphics character) has replaced the first word, FOR, of our first BASIC statement because you put that value into that location. This trick can put characters unavailable from the keyboard into a program. Our Big Characters program (Dec. 80) used this fact to insert the graphics characters to form the big letters (some of which aren't available from the keyboard) into the program in a specific order. To do this yourself, you need to know the first location of the characters you want to insert and begin poking new codes from there on.

POKE commands also allow you to enter machine language codes from BASIC into consecutive memory locations. Because your computer only understands ML commands and data in context, poking a value into a single byte will probably not get you anywhere. Poking a series of values can form a program, as we saw in our analysis of Sinclair's available memory program (May 81).

CLASSIFIEDS

Got something to sell or swap? Or are you looking for something special? Use SYNTAX classifieds. Reach hundreds of other readers for just $2.75 per line (4 line min.). Send your ad, typed 35 characters per line, with payment by 15th of any month for next month's issue to Classified Ads, SYNTAX, RD 2 Box 457 Bolton Rd., Harvard, MA 01451.

Programs-games & utility. also tech data, mods, plans, info, etc. Send SASE for free goodies list P.O. Box 3073 San Jose, CA 95116

Handy Utility Programs
S 002--renumbers BASIC programs, including GO TO / GO SUB.
S 004--formatted core dump, with addr, code, hex, graphics.
S 005--load, display, correct machine language programs.
Written in BASIC. $1.00 each or 3 for $2.50. Martin Irons, 46 Magic Circle Drive, Goshen, NY 10924.

Zilog Z80-CPU Z80A-CPU Technical Manual, $7.50, and Z80-Assembly Language Programming Manual, $15.00. Add 5% for postage and handling. Send check or credit card no. to SYNTAX, RD 2 Box 457 Bolton Rd., Harvard, MA 01451.

Video secrets revealed! This 8-page document explains in detail how ZX80 video works, including a USR routine to run the display. Also included are an explanation and USR routine to run the keyboard and some notes on ROM routines. $4.00 postpaid. Jim Williams, 262 Chappel, Calumet City, IL, 60409.

Nov/Dec SYNTAX combined reprints, $5, some other back issues available, $4 each. Send check credit card no. to SYNTAX, RD 2 Box 457 Bolton Rd., Harvard, MA, 01451.

For many new ZX80 owners, this is your first and only computer. You look to us to help you understand what to do with it.

To help you, we'd like you to help us. We're looking for some beginners to review our beginners' articles each month. Send us your name and address and we'll send you a copy of each beginners' story and a self-addressed stamped envelope. Just read the stories and tell us what does or doesn't help you. No payment, no glory, just better beginners' columns to help you use your computer better. Thanks.

SYNTAX
THE HARVARD GROUP
Bolton Road, Harvard, Mass. 01451

First Class